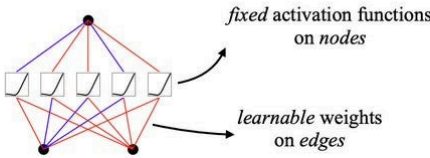
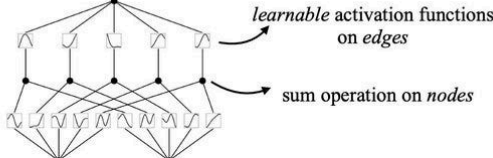
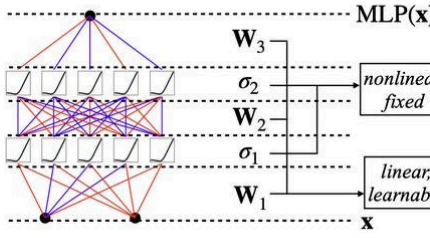
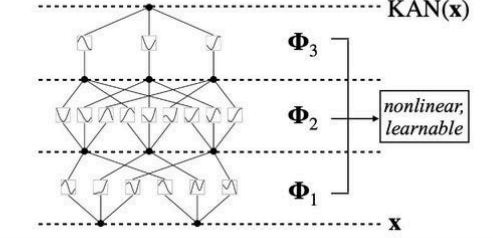


Model	Multi-Layer Perceptron (MLP)	Kolmogorov-Arnold Network (KAN)
Theorem	Universal Approximation Theorem	Kolmogorov-Arnold Representation Theorem
Formula (Shallow)	$f(\mathbf{x}) \approx \sum_{i=1}^{N(e)} a_i \sigma(\mathbf{w}_i \cdot \mathbf{x} + b_i)$	$f(\mathbf{x}) = \sum_{q=1}^{2n+1} \Phi_q \left(\sum_{p=1}^n \phi_{q,p}(x_p) \right)$
Model (Shallow)	(a) 	(b) 
Formula (Deep)	$\text{MLP}(\mathbf{x}) = (\mathbf{W}_3 \circ \sigma_2 \circ \mathbf{W}_2 \circ \sigma_1 \circ \mathbf{W}_1)(\mathbf{x})$	$\text{KAN}(\mathbf{x}) = (\Phi_3 \circ \Phi_2 \circ \Phi_1)(\mathbf{x})$
Model (Deep)	(c) 	(d) 

(위 표에서 KAN은 아래에서 위로 진행된다, width는 [2,3,3,1]로 생각된다)

A screenshot from: <https://arxiv.org/abs/2404.19756>

물리나 수학 학문은 그 어려움으로 인해 인공지능의 도전분야가 되어왔다.

인공지능의 성능의 척도중 하나가 수학풀이(4-shot)이기도 하다. 그런데, KAN이란 뉴럴네트워크의 강력한 특성들은 물리연구중 특정 분야를 가능케할것으로 기대된다.

Kolmogorov_Arnold Network 는 activation function이 고정되어있고 weight값이 변하는 일반적인 MLP 와 달리 KAN은 node에 activation function이 존재하지 않고 edge에 존재하며 이 함수를 learnable 하게 설정하여 해당 함수가 변하며 데이터를 학습하는 네트워크이다. 이때 node는 단순히 input을 합산하는 역할을 한다.

이 KAN의 주요 특징은 위 표에서 볼 수 있듯이 KAN은 non-linear하며 learnable한 성질을 가진다는 것이다. 이런 non-linear learnable 특성덕에 과학분야의 모델링에 쓰일 수 있으며 특히 물리에서 데이터를 통해 complex modeling을 수행할 수 있으며, 시도를 하고 있다.

나는 이 KAN을 여러가지 함수에 fitting하고, parameter를 실험하며, 성능과 한계를 실험 해보려고한다.

데이터는 github.com/pykan 의 create_dataset method를 이용하여 생성한다.

KAN과 옵티마이저 LBFGS를 사용하여 훈련한다.

주요 parameter와 KAN의 훈련 결과를 같이 나열했다.

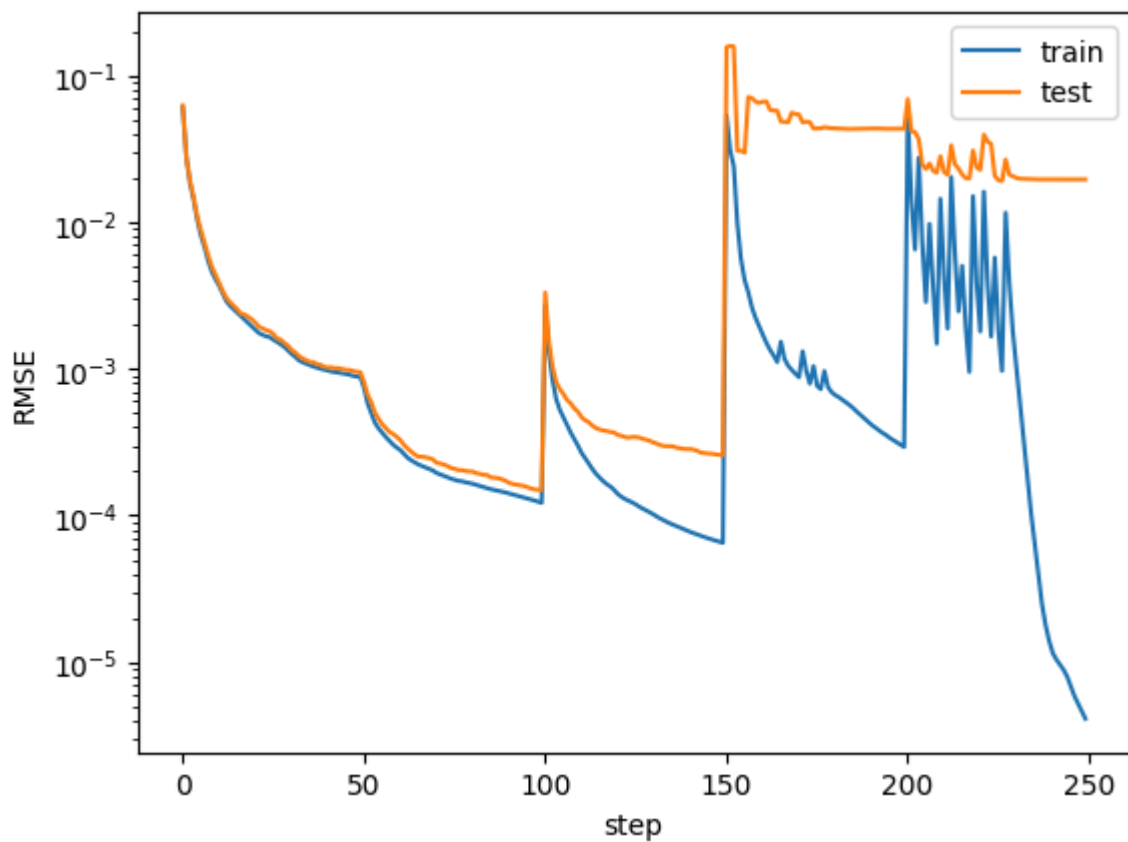
spline function은 activation function을 대신하는 learnable function을 의미하며, grid는 spline function이 곡선함수일때 그 부드러움을 결정한다.

width는 [input_dim, neuron ,output_dim]을 나타낸다.

$\sin(x_0) + \cos(x_1)$ 을 fitting 해보자.

width = [2,10,1]

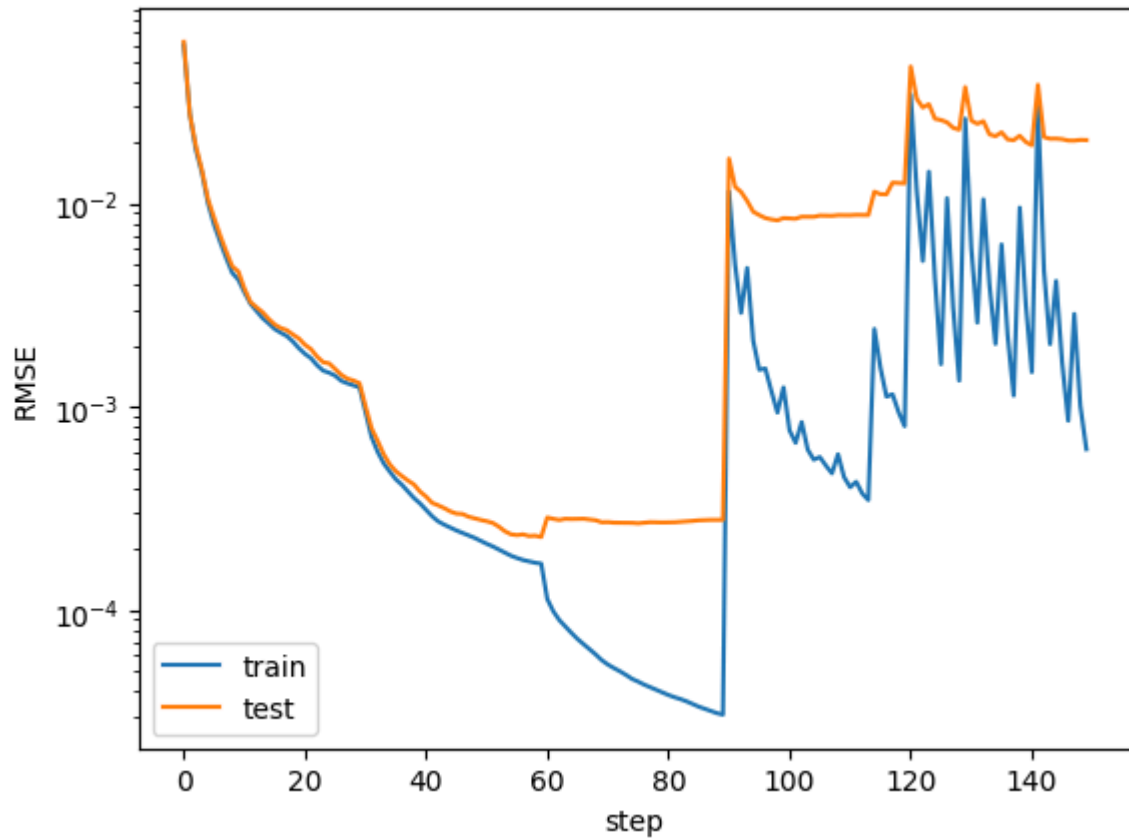
, grid=3 : 20 step, grid = [5,10,20,50,100] : 50 step each



위 문제에서, **grid interval**을 늘려서 20으로 훈련할때부터, 즉 **spline function**이 더 부드럽게 그려질 수 있을때 훈련로스, 테스트로스 모두 악화되는것을 알 수 있다. 이것이 많은 **step**수로 인한 과적화의 문제인지 **spline function** 자체가 과적화 되는것인지는 알 수 없다. 그러므로 **step**을 줄이고 **grid interval**은 유지해서 다시 실험해본다.

width = [2,10,1]

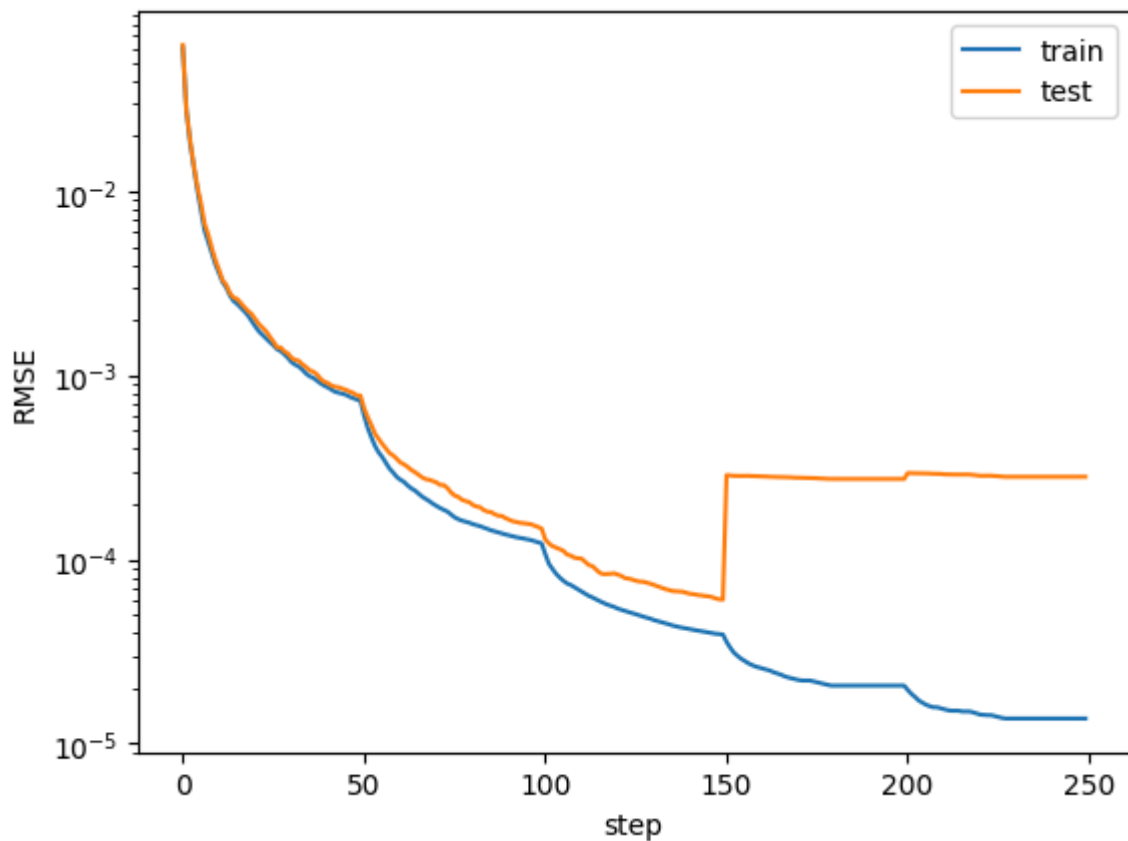
, grid=3 : 20 step, grid = [5,10,20,50,100] : 30 step each



30 step으로 줄일때, **loss** 는 더욱 커지고 마지막 훈련까지 적절히 로스를 줄이지 못함을 알 수 있다. 이는 **step**이 부족함을 의미한다고 볼 수 있다. 그럼 **step**을 50으로 유지후 **grid**를 줄여본다.

width = [2,10,1]

, grid=3 : 20 step, grid = [5,10,15,20,25] : 50 step each

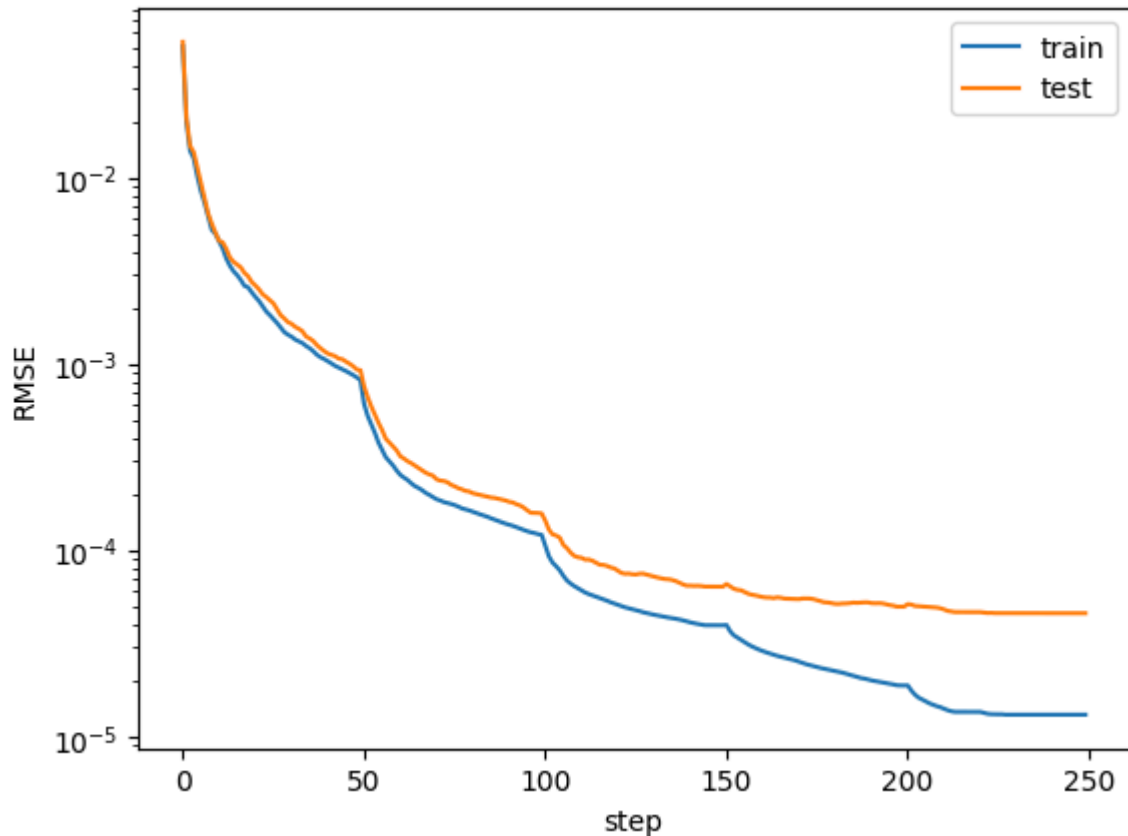


interval 20 이상의 훈련부터는 훈련로스는 낮아지나 테스트 로스는 낮아지지 않는다. 고로 interval 20미만까지 설정하는것이 과적화 문제를 막는것으로 보인다. 이때, spline function이 부드러워지는것이 과적화 문제를 일으키는것은, $\sin(x)+\cos(y)$ 가 fitting을 하기엔 간단하여 interval 20이상의 복잡한 spline function을 필요로 하지 않는것인가 하는 의문이 든다. 그러므로 함수를 더 복잡하게하여 실험해본다.

이 밑으론 $\sin(x)+\cos(y)+\log(1+x^2+y^2)$ 를 fitting 해본다.

width = [2,10,1]

, grid=3 : 20 step, grid = [5,10,15,20,25] : 50 step each



예상대로 **fitting** 하려는 함수가 어려워질수록 **grid interval**이 늘어나는것은 과적화 문제를 일으키지않고 성능을 개선시킨다.

이때 우리는, **KAN**을 함수에 **fitting** 시킬때 함수의 복잡도에 적절한 **grid interval(grid)**이 필요함을 알았다.

이는 이미 알고있는 원함수에 **KAN**을 **fitting**시키는것이 아닌, 미지의 함수를 따른다고 추정되는 임의의 데이터에 **KAN**을 **fitting**시켜 **KAN**을 마치 원함수처럼 이용하려는 시도에서 **parameter** 설정이 난해함을 알 수 있다. 원함수가 얼마나 복잡한지 알 수 없기 때문이다. 특히, 물리나 금융시장의 모델링처럼 원함수가 존재는할지, 랜덤할지 모르는 상황에서선 더욱 그러하다. 하지만 **KAN**의 강력한 특성인 **explainable**한 특성을 이용해 원함수를 **KAN**이 올바르게 **fitting**할때 임의의 **list**에 있는 함수로 원함수를 복원을 시도할 수 있다. 예를들어, 원함수가 $\text{xcos}(x)$ 라면 $[x, \cos(x), \sin(x), \log(x)]$ 를 임의로 정하고 원함수를 **list** 속의 함수들의 조합으로 복원을 시도할 수 있다.

그래서 위의 함수를 복원하려 했으나, **overflow**가 일어났고, 그래서 더 간단한 함수인 $\cos(x) + \sin(y)$ 로 돌아가 복원을 시도했으나, **overflow**는 일어나지 않았지만 원함수와는 전혀다른 매우 복잡한 함수로 복원했다.

그래서 간단한 $\sin(\pi(x))$ 를 복원하기로 하였다.

width = [2,5,1] grid 3 : 20step 이다.

$$1.01 \sin \left(119.52 \sqrt{1 - 0.05x_1} - 116.36 \right) + 0.01$$

$$1.01 \sin \left(94.88 \sqrt{1 - 0.07x_1} - 85.43 \right) + 0.01$$

두번의 복원시도에도 서로 다른 함수를 복원했다.

width = [2,10,1] grid 3 : 20step 시도시에

$$0.76 \sin \left(24.41 \sqrt{0.26x_1 + 1} - 24.47 \right) + 0.42 \sin \left(2.46 \sin \left(1.16x_1 - 7.04 \right) - 4.09 \right) - 0.17 \\ - 0.73 \sin \left(33.93 \sqrt{0.2x_1 + 1} - 30.69 \right) + 0.36 \sin \left(2.56 \sin \left(1.17x_1 + 5.44 \right) + 8.14 \right) - 0.05$$

위의 결과가 나왔다.

neuron 이 과하게 많으면, 그리고 **loss**를 줄이기위해 훈련의 **grid interval**을 늘릴수록 복원된 함수가 더 복잡해지는 경향이 보인다.

이는 임의의 복잡한 함수를 복원가능할지, 그것이 적합할지의 여부는 **concrete**하게 알 수 없음을 의미한다.

결국, 원함수를 모를때 **KAN**의 파라미터 설정의 어려움, 로스를 줄이기 위한 파라미터 설정시 복원된 함수가 과도하게 복잡해지는것, 그리고 파라미터 설정에 따라 복원된 함수의 변화를 감안했을때, 현재의 **KAN**으로는 아직 임의의 데이터에대한 연구는 어렵다고 생각된다. 데이터에 더 나은 **fitting**을 시도하면 원함수가 복원되지 못하고 복원된 함수가 필요 이상으로 복잡해질 수 있기 때문이다.